TECHNICAL REPORT RD-GC-92-33

REAL TIME EXECUTIVE FOR MISSILE SYSTEMS i80386 ASSEMBLY INTERFACE

Wanda M. Hughes and
Phillip R. Acuff
Guidance and Control Directorate
Research, Development, and Engineering Center

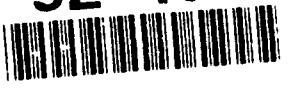
and

On-line Applications Research Corporation
3315 Memorial Parkway SW
Huntsville, AL 35801

MAY 1992

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for Public Release.



DESTRUCTION NOTICE

FOR CLASSIFIED DOCUMENTS, FOLLOW THE PROCEDURES IN DoD 5200.22-M, INDUSTRIAL SECURITY MANUAL, SECTION II-19 OR DoD 5200.1-R, INFORMATION SECURITY PROGRAM REGULATION, CHAPTER IX. FOR UNCLASSIFIED, LIMITED DOCUMENTS, DESTROY BY ANY METHOD THAT WILL PREVENT DISCLOSURE OF CONTENTS OR RECONSTRUCTION OF THE DOCUMENT.

DISCLAIMER

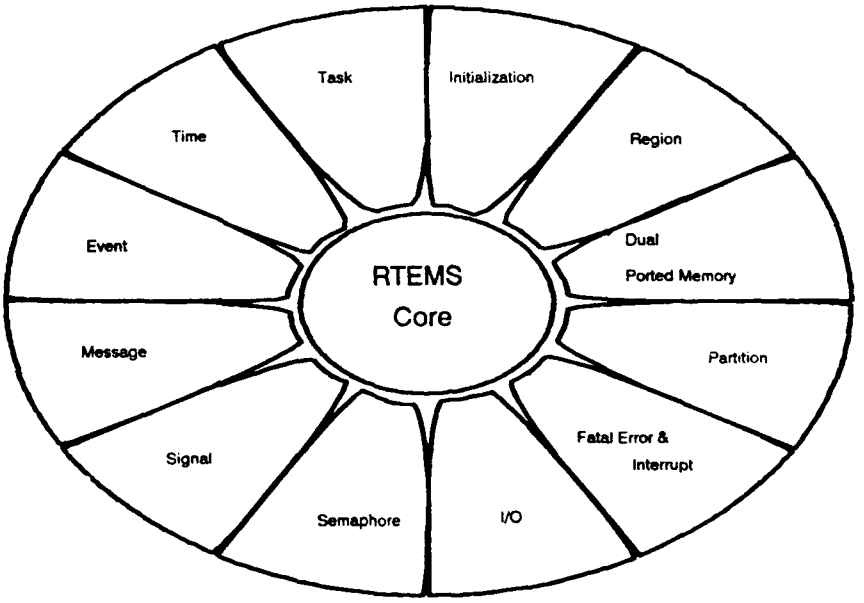
THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

Real Time Executive for Missile Systems

i80386 Assembly Interface



U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5254

Release 1.31
December 1991

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report RD-GC-92-33	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Guidance & Control Directorate RD&E Center 6b. OFFICE SYMBOL (*If applicable*) AMSMI-RD-GC-S	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*) Commander, U.S. Army Missile Command ATTN: AMSMI-RD-GC-S Redstone Arsenal, AL 35898-5254	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE (*Include Security Classification*) Real Time Executive For Missile Systems i80386 Assembly Interface	
12. PERSONAL AUTHOR(S) Wanda M. Hughes, Phillip R. Acuff, and OAR Corp.	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM 6/89 TO 2/92	14. DATE OF REPORT (*Year, Month, Day*) 1992, May 15. PAGE COUNT 82
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
RTEMS, real-time, executive, heterogeneous, homogeneous, multiprocessing, 80386, microprocessor, assembly language,
(Continued on page ii)

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This document details the assembly language interface for the RTEMS (Real-Time Executive for Missile Systems) real-time executive for the Intel 80386 processor. Each entry in this manual corresponds to a directive in RTEMS. Each directive entry details which registers are used for input arguments and return values in addition to giving an example usage. The examples in this document are given in standard Intel 80386 assembly language.

RTEMS is a real-time executive (kernel) which provides a high performance environment for embedded military applications including such features as multitasking capabilities; homogeneous and heterogeneous multiprocessor systems; event-driven, priority-based, preemptive scheduling; intertask communication and synchronization; responsive interrupt management; dynamic memory allocation; and a high level of user configurability. RTEMS was originally developed in an effort to eliminate many of the major drawbacks of the Ada programming
(Continued on page ii)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Wanda M. Hughes	22b. TELEPHONE (*Include Area Code*) (205) 876-4484 22c. OFFICE SYMBOL AMSMI-RD-GC-S

DD Form 1473, JUN 86 *Previous editions are obsolete.*

BLOCK 18 (Cont'd): runtime, directive, multitasking, event-driven, priority-based, preemptive, scheduling, intertask communication, synchronization, dynamic memory allocation, user configurable, kernel, embedded, semaphore, events, interrupt, regions, segments, signals, I/O, messages, user extendable, object oriented

BLOCK 19 (Cont'd): language. RTEMS provides full capabilities for management of tasks, interrupts, time, and multiple processors in addition to those features typical of generic operating systems. The code is Government owned, so no licensing fees are necessary. The executive is written using the 'C' programming language with a small amount of assembly language code. The code was developed as a linkable and/or ROMable library with the Ada programming language. Initially RTEMS was developed for the Motorola 68000 family of processors. It has since been ported to the Intel 80386 and 80960 families. This manual describes the implementation of RTEMS for the i80386 microprocessor for applications using the Ada programming language. Related documents include: Real Time Executive for Missile Systems User's Guide i80386 'C' Interface, Real Time Executive for Missile Systems i80386 Timing Document, and Real Time Executive for Missile Systems i80386 Ada Interface. RTEMS documentation and code is available for the Motorola 68000 family, and the Intel 80386 and 80960 family of processors.

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Table of Contents

S.1 Introduction

S.1.1 Description

This supplemental document contains the assembly language interface for the **RTEMS** real-time executive for the Intel 80386. For more detailed information regarding exact operation, constants, arguments, and data structures, please refer to the manual page for the appropriate directive.

Each entry in this supplemental document corresponds to a directive and details which registers are used for input arguments and return values in addition to an example usage. The examples in this supplement are given in standard Intel 80386 assembly language.

S.1.2 Register Usage

RTEMS-80386 uses the 80386 **EAX**, **ECX**, and **EDX** registers as scratch registers. The contents of these three registers will not be preserved by **RTEMS** directives unless noted otherwise.

S.1.3 Segment Usage

RTEMS-80386 is designed to operate in the thirty-two bit flat memory model of the 80386 with paging disabled. In this mode, the 80386 automatically converts every address from a logical to a physical address each time it is used. The 80386 uses information provided in the segment registers and the **Global Descriptor Table** to accomplish this. **RTEMS-80386** assumes the existence of the following segments:

- *a single code segment at protection level zero (0) which contains all application and executive code.*
- *a single data segment at protection level zero (0) which contains all application and executive data.*

The 80386 segment registers and associated selectors must be initialized when the **init_exec** directive is invoked. **RTEMS-80386** treats the segment registers as system resources and does not modify or context switch them.

RTEMS-80386 does not require that logical and physical addresses are the same, although it is desirable in many applications to do so. If logical and physical addresses are different, the application may require an additional selector to access physical addresses.

S.2 INITIALIZATION MANAGER

S.2.1 INIT_EXEC - Initialize RTEMS

INPUT:

EAX = function code
4[ESP] = address of configuration table

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

push    offset Conf_tbl         ; push address of config table
mov     EAX,INIT_EXEC           ; EAX = function code
call    rtems                   ; enter the executive

; does not return

        .
        .
        .
```

NOTES:

This directive does not return to the caller.

S.3 TASK MANAGER

S.3.1 T_CREATE - Create a task

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = priority
12[ESP] = stack size (in bytes)
16[ESP] = mode
20[ESP] = attributes
24[ESP] = address of task id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

    push    offset Task_id          ; push pointer to task id
    push    TASK_ATTRIBUTES         ; push attributes
    push    TASK_MODE               ; push mode
    push    STACK_SIZE              ; push stack size
    push    PRIORITY                ; push priority
    push    TASK_NAME               ; push name
    mov     EAX,T_CREATE            ; EAX = function code
    call    rtems                   ; enter the executive

    ; should check return code here

        .
        .
        .
```

S.3.2 T_IDENT - Get ID of a task

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = node identifier (defines search space)
12[ESP] = address of task id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Task_id          ; push pointer to task id
push    NODE                    ; push node identifier
push    TASK_NAME               ; push name
mov     T_IDENT,EAX             ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.3 T_START - Start a task

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = entry point
12[ESP] = start argument

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    TASK_ARG            ; push start argument
push    User_task           ; push entry point
push    Task_id             ; push task id
mov     EAX,T_START         ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.4 T_RESTART - Restart a task

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = restart argument

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    RESTART_ARG             ; push restart argument
push    Task_id                 ; push task id
mov     EAX,T_RESTART           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.5 T_DELETE - Delete a task

INPUT:

EAX = function code
4[ESP] = task id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Task_id                 ; push task id
mov     EAX,T_DELETE            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.6 T_SUSPEND - Suspend a task

INPUT:

EAX = function code
4[ESP] = task id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Task_id                 ; push task id
mov     EAX,T_SUSPEND           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.7 T_RESUME - Resume a task

INPUT:

EAX = function code
4[ESP] = task id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Task_id                 ; push task id
mov     EAX,T_RESUME            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.8 T_SETPRI - Set task priority

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = new priority
12[ESP] = address of previous priority storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Prev_priority ; push pointer to previous priority
push    PRIORITY             ; push new priority
push    Task_id              ; push task id
mov     EAX,T_SETPRI         ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.9 T_MODE - Change current task's mode

INPUT:

EAX = function code
4[ESP] = new mode
8[ESP] = mask
12[ESP] = address of previous mode storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .
push    offset Prev_mode        ; push pointer to previous mode
push    MASK                    ; push mask
push    NEW_MODE                ; push new mode
mov     EAX,T_MODE              ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.10 T_GETNOTE - Get task notepad entry

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = notepad entry number
12[ESP] = address of note value storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Note_val         ; push pointer to note value
push    NOTE_NUM                ; push entry number
push    Task_id                 ; push task id
mov     EAX,T_GETNOTE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.3.11 T_SETNOTE - Set task notepad entry

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = notepad entry number
12[ESP] = note value

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    NOTE_VALUE          ; push note value
push    NOTE_NUM            ; push entry number
push    Task_id             ; push task id
mov     EAX,T_SETNOTE       ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.4 INTERRUPT MANAGER

S.4.1 I_ENTER - Enter an ISR

INPUT:

EAX = function code

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

push    EAX                 ; save task's EAX
mov     EAX,I_ENTER         ; EAX = function code
call    rtems               ; enter the executive

; no need to check the return code here

        .
        .
        .
```

NOTES:

This directive uses the **EAX** register only. This register must be saved by the application before invoking **I_ENTER**. The **EAX** register is restored automatically by the **I_RETURN** directive.

S.4.2 I_RETURN - Return from an ISR

INPUT:

EAX = function code

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

mov     EAX,I_RETURN          ; EAX = function code
call    rtems                 ; enter the executive

; will never return

        .
        .
        .
```

NOTES:

This directive uses the **EAX** only. It restores **EAX** to its contents prior to invoking **I_ENTER**.

This directive does not return to the caller.

S.5 TIME MANAGER

S.5.1 TM_SET - Set system date and time

INPUT:

EAX = function code
4[ESP] = address of time_info data structure

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Time_str         ; push pointer to time buffer
mov     EAX,TM_SET              ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.2 TM_GET - Get system date and time

INPUT:

EAX = function code
4[ESP] = address of time_info data structure

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Time_str      ; push pointer to time buffer
mov     EAX,TM_GET           ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.3 TM_WKAFTER - Wake up after interval

INPUT:

EAX = function code
4[ESP] = length of interval (in ticks)

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    INTERVAL                ; push ticks to wait
mov     EAX,TM_WKAFTER          ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.4 TM_WKWHEN - Wake up when specified

INPUT:

EAX = function code
4[ESP] = address of time_info data structure

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Time_str         ; push time to wake
mov     EAX,TM_WKWHEN           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.5 TM_EVAFTER - Send event set after interval

INPUT:

EAX = function code
4[ESP] = interval until event (in ticks)
8[ESP] = event set
12[ESP] = address of timer id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Timer_id         ; push pointer to timer id
push    EVENTS                  ; push events to send
push    INTERVAL                ; push ticks until event
mov     EAX,TM_EVAFTER          ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.6 TM_EVWHEN - Send event set when specified

INPUT:

EAX = function code
4[ESP] = address of time_info data structure
8[ESP] = event set
12[ESP] = address of timer id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Timer_id      ; push pointer to timer id
push    EVENTS               ; push events to send
push    offset Time_str      ; push time to send events
mov     EAX,TM_EVWHEN        ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.7 TM_EVEVERY - Send periodic event set

INPUT:

EAX = function code
4[ESP] = interval between events (in ticks)
8[ESP] = event set
12[ESP] = address of timer id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Timer_id      ; push pointer to timer id
push    EVENTS               ; push events to send
push    NUM_TICKS            ; push time between events
mov     EAX,TM_EVEVERY       ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
        .
```

S.5.8 TM_CANCEL - Cancel timer event

INPUT:

EAX = function code
4[ESP] = timer event id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     Timer_id              ; push timer id
     mov      EAX,TM_CANCEL         ; EAX = function code
     call     rtems                 ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.5.9 TM_TICK - Announce a clock tick

INPUT:

EAX = function code

OUTPUT:

EAX = SUCCESSFUL

EXAMPLE:

```
          .
          .
          .

     mov       EAX,TM_TICK          ; EAX = function code
     call      rtems                ; enter the executive

     ; no need to check the return code here

          .
          .
          .
```

S.6 SEMAPHORE MANAGER

S.6.1 SM_CREATE - Create a semaphore

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = initial count
12[ESP] = attributes
16[ESP] = address of semaphore id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Sem_id           ; push pointer to semaphore id
push    SEM_ATTRIBUTES          ; push attributes
push    INITIAL_COUNT           ; push initial count
push    SEM_NAME                ; push name
mov     EAX,SM_CREATE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.6.2 SM_IDENT - Get ID of a semaphore

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = node identifier (defines search space)
12[ESP] = address of semaphore id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Sem_id          ; push pointer to semaphore id
push    NODE                   ; push node identifier
push    SEM_NAME               ; push name
mov     EAX,SM_IDENT           ; EAX = function code
call    rtems                  ; enter the executive

; should check return code here

        .
        .
        .
```

S.6.3 SM_DELETE - Delete a semaphore

INPUT:

EAX = function code
4[ESP] = semaphore id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Sem_id                  ; push semaphore id
mov     EAX,SM_DELETE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.6.4 SM_P - Acquire a semaphore

INPUT:

EAX = function code
4[ESP] = semaphore id
8[ESP] = options
12[ESP] = maximum interval to wait (in ticks)

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    TIMEOUT             ; push maximum ticks to wait
push    OPTIONS             ; push options
push    Sem_id              ; push semaphore id
mov     EAX,SM_P            ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.6.5 SM_V - Release a semaphore

INPUT:

EAX = function code
4[ESP] = semaphore id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     Sem_id                ; push semaphore id
     mov      EAX,SM_V              ; EAX = function code
     call     rtems                 ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.7 MESSAGE MANAGER

S.7.1 Q_CREATE - Create a queue

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = maximum message count
12[ESP] = attributes
16[ESP] = address of queue id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Queue_id         ; push pointer to queue id
push    Q_ATTRIB                ; push attributes
push    MSG_BUF_COUNT           ; push message count
push    QUEUE_NAME              ; push name
mov     EAX,Q_CREATE            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.2 Q_IDENT - Get ID of a queue

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = node identifier (defines search space)
12[ESP] = address of queue id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Queue_id         ; push pointer to queue id
push    NODE                    ; push node identifier
push    QUEUE_NAME              ; push name
mov     EAX,Q_IDENT             ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.3 Q_DELETE - Delete a queue

INPUT:

EAX = function code
4[ESP] = queue id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Queue_id            ; push queue id
mov     EAX,Q_DELETE        ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.4 Q_SEND - Put message at rear of a queue

INPUT:

EAX = function code
4[ESP] = queue id
8[ESP] = address of message buffer

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Message          ; push address of message
push    Queue_id                ; push queue id
mov     EAX,Q_SEND              ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.5 Q_URGENT - Put message at front of a queue

INPUT:

EAX = function code
4[ESP] = queue id
8[ESP] = address of message buffer

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Message          ; push address of message
push    Queue_id                ; push queue id
mov     EAX,Q_URGENT            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.6 Q_BROADCAST - Broadcast N messages to a queue

INPUT:

EAX = function code
4[ESP] = queue id
8[ESP] = address of message buffer
12[ESP] = address of "number of tasks made ready" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     offset Num_tasks      ; push pointer to number
                                    ;   of tasks readied
     push     offset Message        ; push address of message
     push     Queue_id              ; push queue id
     mov      EAX,Q_BROADCAST       ; EAX = function code
     call     rtems                 ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.7.7 Q_RECEIVE - Receive message from a queue

INPUT:

EAX = function code
4[ESP] = queue id
8[ESP] = address of message buffer
12[ESP] = options
16[ESP] = maximum interval to wait (in ticks)

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    TIMEOUT                 ; push maximum ticks to wait
push    OPTIONS                 ; push receive options
push    offset Message          ; push pointer to message
push    Queue_id                ; push queue id
mov     EAX,Q_RECEIVE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.7.8 Q_FLUSH - Flush all messages on a queue

INPUT:

EAX = function code
4[ESP] = queue id
8[ESP] = address of "number of messages flushed" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     offset Num_flushed    ; push pointer to number
                                    ;   of messages flushed
     push     Queue_id              ; push queue id
     mov      EAX,Q_FLUSH           ; EAX = function code
     call     rtems                 ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.8 EVENT MANAGER

S.8.1 EV_SEND - Send event set to a task

INPUT:

EAX = function code
4[ESP] = task id to send events to
8[ESP] = event set to send

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    EVENTS                  ; push events to send
push    Task_id                 ; push task id
mov     EAX,EV_SEND             ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.8.2 EV RECEIVE - Receive event condition

INPUT:

EAX = function code
4[ESP] = input event condition
8[ESP] = options
12[ESP] = maximum interval to wait (in ticks)
16[ESP] = address of events received storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Events_recvd  ; push pointer to events received
push    TIMEOUT              ; push maximum ticks to wait
push    OPTIONS              ; push receive options
push    EVENTS               ; push event condition
mov     EAX,EV_RECEIVE       ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
        .
```

S.9 SIGNAL MANAGER

S.9.1 AS_CATCH - Establish an ASR

INPUT:

EAX = function code
4[ESP] = address of ASR
8[ESP] = mode of ASR

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    ASR_MODE            ; push ASR mode
push    offset Asr          ; push ASR address
mov     EAX,AS_CATCH        ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.9.2 AS_SEND - Send signal set to a task

INPUT:

EAX = function code
4[ESP] = task id
8[ESP] = signal set

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    SIGNALS                 ; push signals to send
push    Task_id                 ; push task id
mov     EAX,AS_SEND             ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.9.3 AS_ENTER - Enter an ASR

INPUT:

EAX = function code

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

push    EAX                     ; save task's EAX
mov     EAX,AS_ENTER            ; EAX = function code
call    rtems                   ; enter the executive

; no need to check the return code here

        .
        .
        .
```

NOTES:

This directive uses the **EAX** register only. This register must be saved by the application before invoking **AS_ENTER**. The **EAX** register is restored automatically by the **AS_RETURN** directive.

S.9.4 AS_RETURN - Return from an ASR

INPUT:

EAX = function code

OUTPUT:

D0 = directive status code

EXAMPLE:

```
          .
          .
          .

     mov       EAX,AS_RETURN          ; EAX = function code
     call      rtems                  ; enter the executive

     ; does not return if SUCCESSFUL

          .
          .
          .
```

NOTES:

This directive uses the **EAX** only. It restores **EAX** to its contents prior to invoking **I_ENTER**.

If successful, this directive does not return to the caller.

S.10 PARTITION MANAGER

S.10.1 PT_CREATE - Create a partition

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = physical start address of partition
12[ESP] = length (in bytes)
16[ESP] = size of buffers (in bytes)
20[ESP] = attributes
24[ESP] = address of partition id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Part_id          ; push pointer to partition id
push    PART_ATTRIBUTES         ; push attributes
push    BUF_SIZE                ; push buffer size
push    PART_LENGTH             ; push length
push    PART_ADDR               ; push start address
push    PART_NAME               ; push name
mov     EAX,PT_CREATE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.10.2 PT_IDENT - Get ID of a partition

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = node identifier (defines search space)
12[ESP] = address of partition id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Part_id          ; push pointer to partition id
push    NODE                    ; push node identifier
push    PART_NAME               ; push name
mov     EAX,PT_IDENT            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.10.3 PT_DELETE - Delete a partition

INPUT:

EAX = function code
4[ESP] = partition id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Part_id             ; push partition id
mov     EAX,PT_DELETE       ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.10.4 PT_GETBUF - Get buffer from a partition

INPUT:

EAX = function code
4[ESP] = partition id
8[ESP] = address of "buffer address" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Buff_addr      ; push pointer to buffer address
push    Part_id               ; push partition id
mov     EAX,PT_GETBUF         ; EAX = function code
call    rtems                 ; enter the executive

 ; should check return code here

        .
        .
        .
```

S.10.5 PT_RETBUF - Return buffer to a partition

INPUT:

EAX = function code
4[ESP] = partition id
8[ESP] = buffer start address

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Buff_addr           ; push buffer address
push    Part_id             ; push partition id
mov     EAX,PT_RETBUF       ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.11 REGION MANAGER

S.11.1 RN_CREATE - Create a region

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = physical start address of region
12[ESP] = length (in bytes)
16[ESP] = page size (in bytes)
20[ESP] = attributes
24[ESP] = address of region id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Regn_id          ; push pointer to region id
push    REGN_ATTRIB             ; push attributes
push    REGN_PAGE               ; push page size
push    REGN_LENGTH             ; push length
push    REGN_ADDRESS            ; push physical starting address
push    REGN_NAME               ; push name
mov     EAX,RN_CREATE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.11.2 RN_IDENT - Get ID of a region

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = address of region id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Regn_id          ; push pointer to region id
push    REGN_NAME               ; push name
mov     EAX,RN_IDENT            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.11.3 RN_DELETE - Delete a region

INPUT:

EAX = function code
4[ESP] = region id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Regn_id             ; push region id
mov     EAX,RN_DELETE       ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.11.4 RN_GETSEG - Get segment from a region

INPUT:

EAX = function code
4[ESP] = region id
8[ESP] = segment size desired (in bytes)
12[ESP] = options
16[ESP] = maximum interval to wait (in ticks)
20[ESP] = address of "segment address" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Seg_addr     ; push pointer to segment address
push    TIMEOUT             ; push maximum ticks to wait
push    OPTIONS             ; push getseg options
push    SEG_SIZE            ; push desired segment size
push    Regn_id             ; push region id
mov     EAX,RN_GETSEG       ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.11.5 RN_RETSEG - Return segment to a region

INPUT:

EAX = function code
4[ESP] = region id
8[ESP] = segment address

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     Seg_addr              ; push segment address
     push     Regn_id               ; push region id
     mov      EAX,RN_RETSEG         ; EAX = function code
     call     rtems                 ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.12 DUAL-PORTED MEMORY MANAGER

S.12.1 DP_CREATE - Create a port

INPUT:

EAX = function code
4[ESP] = user-defined four byte name
8[ESP] = starting internal address
12[ESP] = starting external address
16[ESP] = length (in bytes)
20[ESP] = address of port id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Port_id          ; push pointer to port id
push    PORT_LENGTH             ; push length of DPMA
push    External_addr           ; push external address
push    Internal_addr           ; push internal address
push    PORT_NAME               ; push name
mov     EAX,DP_CREATE           ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.12.2 DP_IDENT - Get ID of a port

INPUT:

EAX = function code
4[ESP] = user-defined name to search for
8[ESP] = address of port id storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Port_id          ; push pointer to port id
push    PORT_NAME               ; push name
mov     EAX,DP_IDENT            ; EAX = function code
call    rtems                   ; enter the executive

; should check return code here

        .
        .
        .
```

S.12.3 DP_DELETE - Delete a port

INPUT:

EAX = function code
4[ESP] = port id

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    Port_id             ; push port id
mov     EAX, DP_DELETE      ; EAX = function code
call    rtems               ; enter the executive

; should check return code here

        .
        .
        .
```

S.12.4 DP_2INTERNAL - Convert external to internal address

INPUT:

EAX = function code
4[ESP] = port id
8[ESP] = external address
12[ESP] = address of "internal address" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
          .
          .
          .

     push     offset Internal_addr ; push pointer to internal address
     push     External_addr        ; push external address
     push     Port_id              ; push port id
     mov      EAX,DP_2INTERNAL     ; EAX = function code
     call     rtems                ; enter the executive

     ; should check return code here

          .
          .
          .
```

S.12.5 DP_2EXTERNAL - Convert internal to external address

INPUT:

EAX = function code
4[ESP] = port id
8[ESP] = internal address
12[ESP] = address of "external address" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset External_addr ; push pointer to external address
push    Internal_addr        ; push internal address
push    Port_id              ; push port id
mov     EAX,DP_2EXTERNAL     ; EAX = function code
call    rtems                ; enter the executive

; should check return code here

        .
        .
```

S.13 INPUT/OUTPUT MANAGER

S.13.1 DE_INIT - Initialize a device driver

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_INIT           ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.13.2 DE_OPEN - Open a device

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .
push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_OPEN           ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.13.3 DE_CLOSE - Close a device

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_CLOSE          ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.13.4 DE_READ - Read from a device

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_READ           ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.13.5 DE_WRITE - Write to a device

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_WRITE          ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.13.6 DE_CNTRL - Special device services

INPUT:

EAX = function code
4[ESP] = device number
8[ESP] = address of parameter block
12[ESP] = address of "return code from device driver" storage location

OUTPUT:

EAX = directive status code

EXAMPLE:

```
        .
        .
        .

push    offset Driver_rval    ; push pointer to driver's
                              ;   return code
push    offset Param_blk      ; push pointer to parameter block
push    DEV_NUM               ; push device number
mov     EAX,DE_CNTRL          ; EAX = function code
call    rtems                 ; enter the executive

; should check directive's return code here
; should check device driver's return code here

        .
        .
        .
```

S.14 FATAL MANAGER

S.14.1 K_FATAL - Invoke the fatal error handler

INPUT:

EAX = function code
4[ESP] = error code

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

    push    Fatal_error          ; push error code
    mov     EAX,K_FATAL          ; EAX = function code
    call    rtems                ; enter the executive

    ; will never return

        .
        .
        .
```

NOTES:

This directive does not return to the caller.

S.15 MULTIPROCESSING MANAGER

S.15.1 MP_ANNOUNCE - Announce the arrival of a packet

INPUT:

EAX = function code

OUTPUT:

NONE

EXAMPLE:

```
        .
        .
        .

mov     EAX,MP_ANNOUNCE       ; EAX = function code
call    rtems                 ; enter the executive

; no need to check the return code here

        .
        .
        .
```

S.16 directives.eq

```
;*****************************************************************
;
;  directives.eq
;
;  The following definitions are the directive numbers used
;  in the assembly interface.
;

INIT_EXEC       EQU     0
I_ENTER         EQU     1
I_RETURN        EQU     2
K_FATAL         EQU     3
TM_SET          EQU     4
TM_GET          EQU     5
TM_WKAFTER      EQU     6
TM_WKWHEN       EQU     7
TM_EVAFTER      EQU     8
TM_EVWHEN       EQU     9
TM_EVEVERY      EQU     10
TM_CANCEL       EQU     11
TM_TICK         EQU     12
T_CREATE        EQU     13
T_IDENT         EQU     14
T_START         EQU     15
T_RESTART       EQU     16
T_DELETE        EQU     17
T_SUSPEND       EQU     18
T_RESUME        EQU     19
T_SETPRI        EQU     20
T_MODE          EQU     21
T_GETNOTE       EQU     22
T_SETNOTE       EQU     23
EV_SEND         EQU     24
EV_RECEIVE      EQU     25
AS_CATCH        EQU     26
AS_SEND         EQU     27
AS_ENTER        EQU     28
AS_RETURN       EQU     29
Q_CREATE        EQU     30
Q_IDENT         EQU     31
Q_DELETE        EQU     32
Q_SEND          EQU     33
Q_URGENT        EQU     34
Q_BROADCAST     EQU     35
Q_RECEIVE       EQU     36
```

```
Q_FLUSH             EQU     37
SM_CREATE           EQU     38
SM_IDENT            EQU     39
SM_DELETE           EQU     40
SM_P                EQU     41
SM_V                EQU     42
RN_CREATE           EQU     43
RN_IDENT            EQU     44
RN_DELETE           EQU     45
RN_GETSEG           EQU     46
RN_RETSEG           EQU     47
PT_CREATE           EQU     48
PT_IDENT            EQU     49
PT_DELETE           EQU     50
PT_GETBUF           EQU     51
PT_RETBUF           EQU     52
DP_CREATE           EQU     53
DP_IDENT            EQU     54
DP_DELETE           EQU     55
DP_2INTERNAL        EQU     56
DP_2EXTERNAL        EQU     57
MP_ANNOUNCE         EQU     58
DE_INIT             EQU     59
DE_OPEN             EQU     60
DE_CLOSE            EQU     61
DE_READ             EQU     62
DE_WRITE            EQU     63
DE_CNTRL            EQU     64

BEGIN_CODE_DCL
  EXTRN rtems:near          ; single RTEMS entry point
END_CODE_DCL

;  end of directives.eq
;
;*****************************************************************
```

S.17 dirstatus.eq

```
;*************************************************************
;
;  dirstatus.eq
;
;  This include file contains the status codes returned
;  from the executive's directives.
;
SUCCESSFUL        EQU        0    ; successful completion
E_EXITTED         EQU        1    ; returned from a task
E_NOMP            EQU        2    ; mp not configured
E_NAME            EQU        3    ; invalid object name
E_ID              EQU        4    ; invalid object id
E_TOOMANY         EQU        5    ; too many
E_TIMEOUT         EQU        6    ; timed out waiting
E_DELETE          EQU        7    ; object deleted while waiting
E_SIZE            EQU        8    ; specified size was invalid
E_ADDRESS         EQU        9    ; address specified is invalid
E_NUMBER          EQU       10    ; number was invalid
E_NOTDEFINED      EQU       11    ; item has been initialized
E_INUSE           EQU       12    ; resources still outstanding
E_UNSATISFIED     EQU       13    ; request not satisfied
E_STATE           EQU       14    ; task is in wrong state
E_ALREADY         EQU       15    ; task already in state
E_SELF            EQU       16    ; illegal on calling task
E_REMOTE          EQU       17    ; illegal on remote object
E_CALLED          EQU       18    ; called from wrong environment
E_PRIORITY        EQU       19    ; invalid task priority
E_CLOCK           EQU       20    ; invalid date/time
E_NODE            EQU       21    ; invalid node id
E_NOTCONFIGURED   EQU       22    ; directive not configured
E_NOTIMPLEMENTED  EQU       23    ; directive not implemented

; end of dirstatus.eq
;
;*************************************************************
```

INITIAL DISTRIBUTION

	Copies
U.S. Army Materiel System Analysis Activity ATTN: AMXSY-MP (Herbert Cohen) Aberdeen Proving Ground, MD 21005	1
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
WL/MNAG ATTN: Chris Anderson Eglin AFB, FL 32542-5434	1
Naval Weapons Center Missile Software Technology Office Code 3901C, ATTN: Mr. Carl W. Hall China Lake, CA 93555-6001	1
On-line Applications Research 3315 Memorial Parkway, SW Huntsville, AL 35801	3
CEA Incorporated Blue Hills Office Park 150 Royall Street Suite 260, ATTN: Mr. John Shockro Canton, MA 01021	1
VITA 10229 N. Scottsdale Rd Suite B, ATTN: Mr. Ray Alderman Scottsdale, AZ 85253	1
Westinghouse Electric Corp. P.O. Box 746 - MS432 ATTN: Mr. Eli Solomon Baltimore, MD 21203	1
Dept. of Computer Science B-173 Florida State University ATTN: Dr. Ted Baker Tallahassee, FL 32306-4019	1
DSD Laboratories 75 Union Avenue ATTN: Mr. Roger Whitehead Studbury, MA 01776	1

	Copies
AMSMI-RD	1
AMSMI-RD-GS, Dr. Paul Jacobs	1
AMSMI-RD-GC-S, Gerald E. Scheiman	1
Wanda M. Hughes	5
Phillip Acuff	4
AMSMI-RD-BA	1
AMSMI-RD-BA-C3, Bob Christian	1
AMSMI-RD-SS	1
AMSMI-RD-CS-R	15
AMSMI-RD-CS-T	1
AMSMI-GC-IP, Mr. Fred M. Bush	1
CSSD-CR-S, Mr. Frank Poslajko	1
SFAE-FS-ML-TM, Mr. Frank Gregory	1
SFAE-AD-ATA-SE, Mr. Julian Cothran	1
Mr. John Carter	1

☆ U. S. GOVERNMENT PRINTING OFFICE 1992 – 631 -100 /60221